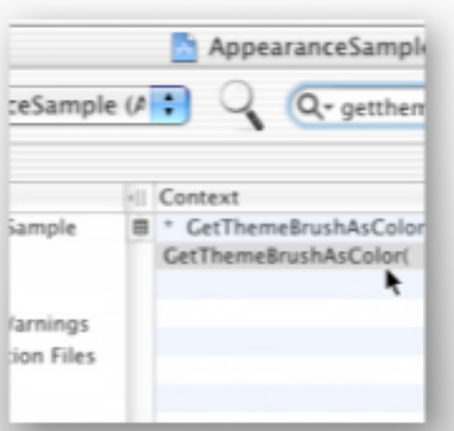
Apple Tools for Carbon Developers

Session 307

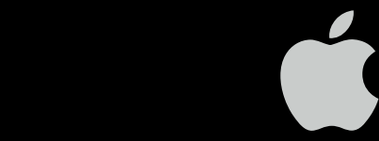


Apple Tools for Carbon Developers

Chris Espinosa
Developer Tools Team

What We'll Cover

- Overview of Apple Tools
- Process of bringing a Project into Xcode
- Getting your code to build with Xcode/GCC
- Optimizing Build and turnaround time
- Xcode's support for Mac Resources
- Designing Carbon Interfaces with Interface Builder



Apple Tools for Carbon Development

Apple Tools for Carbon Development

Apple Tools for Carbon Development

- Xcode integrated development environment
  - Project manager and build system
  - Code editor
  - Debugger

Apple Tools for Carbon Development

- Xcode integrated development environment
  - Project manager and build system
  - Code editor
  - Debugger

- Interface Builder

Apple Tools for Carbon Development

- Xcode integrated development environment
  - Project manager and build system
  - Code editor
  - Debugger

- Interface Builder
- Performance tools
  - Sampler performance analysis tool
  - MallocDebug memory analyzer
  - ObjectAlloc object analyzer for Objective-C
  - CHUD low-level machine-code analysis tools

Other Tools Xcode Works With

- Editors
  - BBEdit
  - emacs
- Design Tools
  - Constructor
  - Resourceror
- Source Control Systems
  - CVS
  - Perforce

Xcode Loves Carbon!

- Easy to import Carbon projects
- Compiles Mac resource files
- Ties in with existing resource editors
- Supports building NIB files for Carbon HI
  - Best way to use Carbon events!
- Supports combining Carbon and Cocoa in the same app

Bringing a Project Into Xcode

Bringing a Project Into Xcode

Steps you need to take

- Carbonize
- Convert to Mach-O-Plug-ins/bundles
- Import into Xcode
- Adjust build settings to get project to build
- Adjust source code to adapt code to gcc

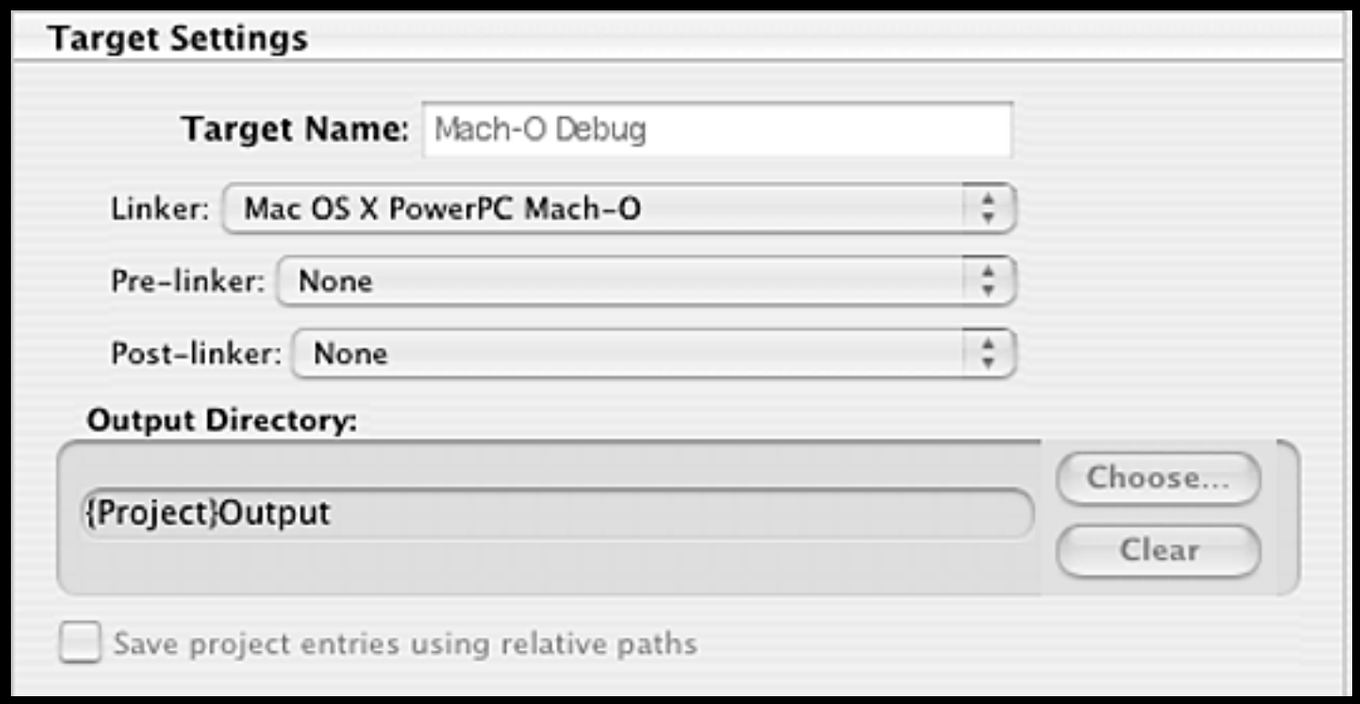
Carbonizing Your Application

You should have done this years ago

- Isolate your Mac OS usage to CarbonLib subset
- Modernize to use CarbonLib 1.6-class APIs
- Remove linkage to InterfaceLib and other stubs
  - Link to CarbonLib (or its stub) instead

Converting to Mach-O

- Mach-O is the native format of Mac OS X
- It's easiest to convert in CodeWarrior first



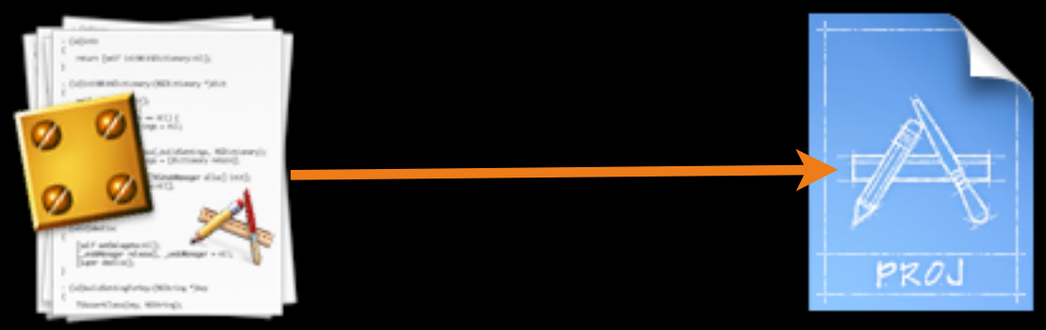
Converting to Mach-O

- Migrate your plugin architecture to CFPlugin
  - Plugins can contain CFM and/or Mach-O
  - Your application can be architecture-neutral
- To support legacy CFM plugins
  - You can call CFM APIs from your Mach-O application

Importing Project Builder Projects

Xcode reads and converts .pbproj files

- JAM-based targets work with Project Builder 2.1
- If you convert to native targets, project is Xcode only



Importing CodeWarrior Projects

Not automatic—but easier than you think

- Import your projects “bottom-up”
- Change to gcc precompiled headers
- Change your PowerPlant sources (one time only)

- Adjust target settings to fix link errors
- Fix resource files to work around Xcode DR bugs
- Work on warnings and errors

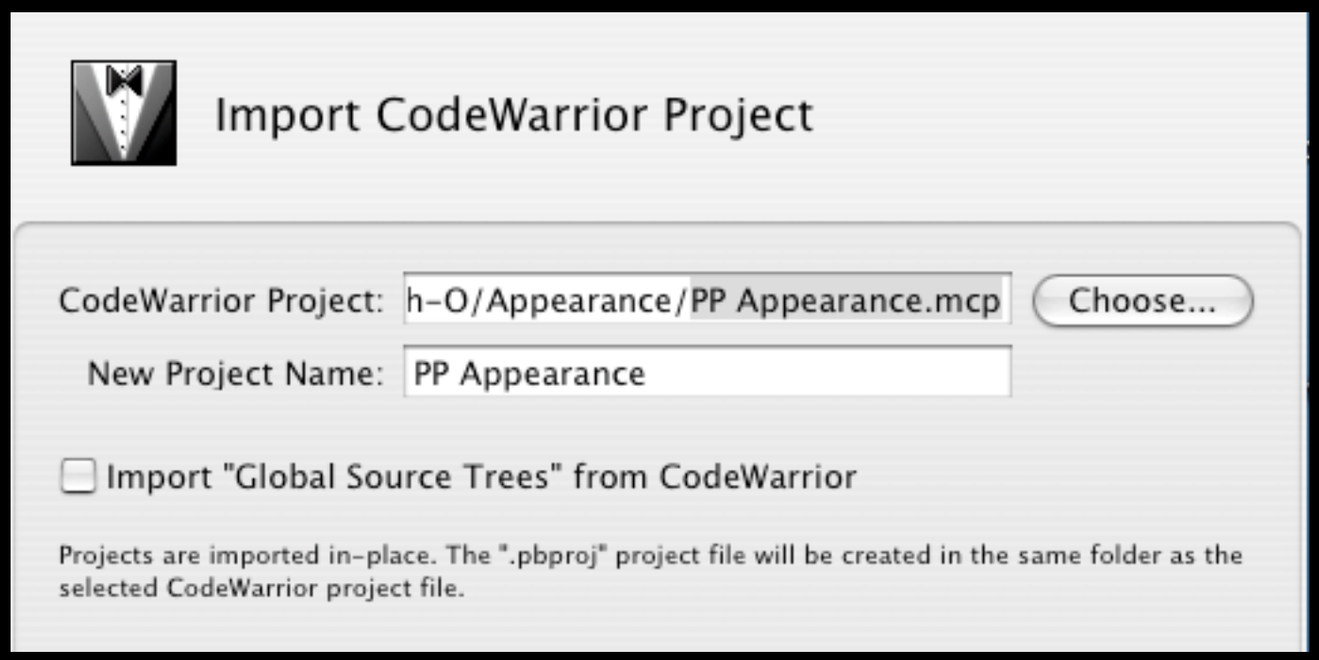
Demo

Importing a Project Into Xcode

Scott Tooker

How the Importer Works

- Sends AppleScript to CW to export project as XML with relative paths
- Imports resulting XML project file into Xcode
- Walks #include paths to find all project header files
- Compiles any .plc files into Info.plist form

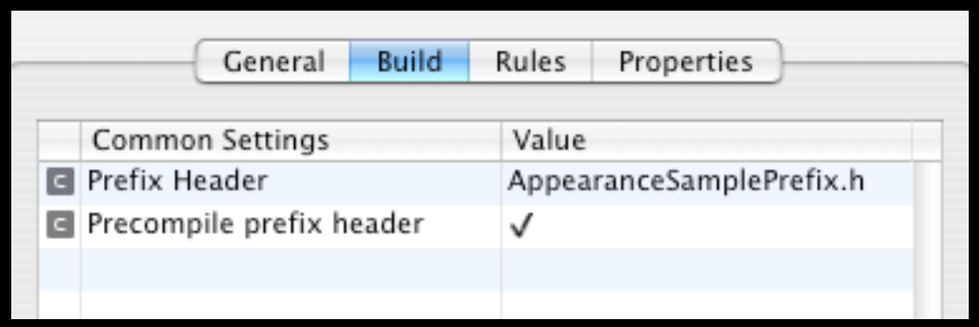


What the Importer Does Not Handle

- Metrowerks precompiled header binaries
- Non-source files in your project tree
- CFM libraries or stubs in your targets
- Inter-target dependencies

Using Precompiled Headers In Xcode

- CodeWarrior precompiles anything
  - .h, .pch, even .c and .cpp
  - Use #pragmas to specify destination file
  - Use a precompiled .mch file interchangeably with .h
  - Use null targets (with throwaway binaries) to precompile
- Xcode only precompiles prefix files—automatically



Demo

Setting Up for Xcode Precompiled Headers

Building PowerPlant With Xcode

PowerPlant is gcc friendly

- Changes to LException.h and LException.cp
  - Add throw() specializers to destructor and what()

Building PowerPlant With Xcode

PowerPlant is gcc friendly

- Changes to LException.h and LException.cp
  - Add throw() specializers to destructor and what()
- Changes to LGATabsControlImp.cp
  - Move local declaration outside of case scope

Building PowerPlant With Xcode

PowerPlant is gcc friendly

- Changes to LException.h and LException.cp
  - Add throw() specializers to destructor and what()
- Changes to LGATabsControlImp.cp
  - Move local declaration outside of case scope
- Changes to LStream.h
  - Omit definitions of operator << and operator >> for long double and short double

Demo

A PowerPlant Application In Xcode

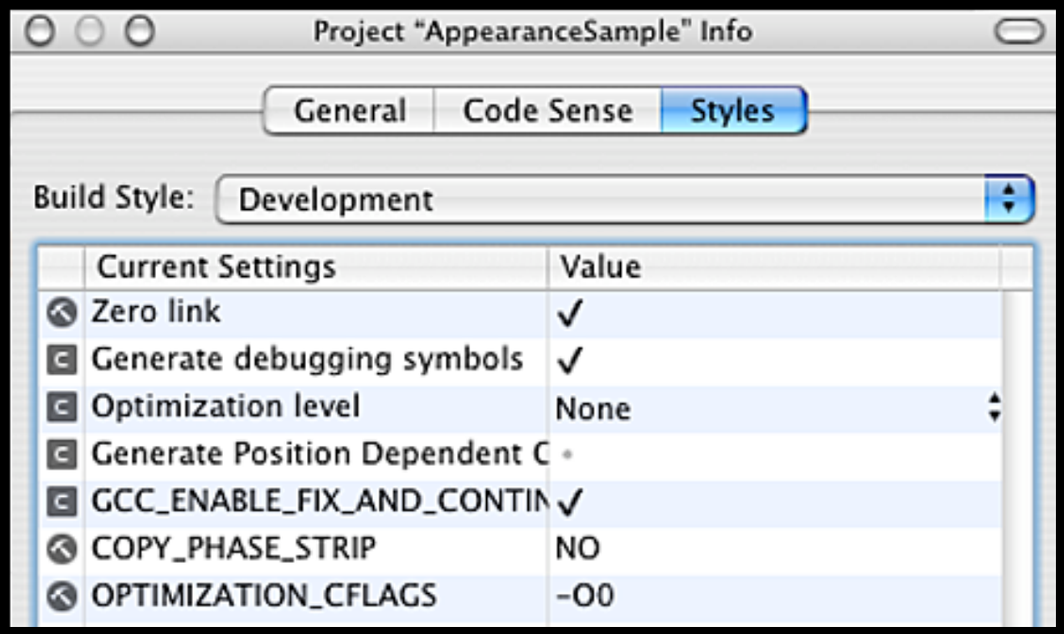
Linking Carbon Targets

- The Importer maps key target settings
- You'll need to set up certain other settings
  - -lstdc++
- Make sure to remove unneeded libraries
  - MSL .a files
- Try a development build to see all your link errors
  - Zero Link hides many interesting errors

Linking Carbon Targets

Build Styles help reduce target proliferation

- Use Development and Deployment styles instead of debug targets
- Project-wide Build Styles affect all targets
- Make one change uniformly across all targets!



Demo

Fixing Link Settings

Moving Resources to .r Files
(This will be fixed by Xcode first release)

- Resources should go into data-fork resource files
- Current version of ResMerger doesn't like mixed resource files
- For now you can just DeRez your .rsrc files into .r

Demo

Fixing Resources and Running

Getting Your Code to Build With Xcode/GCC

Your Code and GCC

Differences between mwcc and gcc

- Very little #pragma support in gcc
- Warnings are very different
- Inlining threshhold is by size, not depth

Your Code and GCC

Differences between mwcc and gcc

- gcc doesn't like C++ code in .c files
  - You can override with `-x c c++`
  - Indexer and debugger will remain confused
- gcc differs where standards are undefined
  - Order of calling C++ static initializers
  - Passing structs/unions of fewer than 4 bytes

Your Code and GCC

Apple Standard Libraries and MSL

- MSL Mach-O lives on top of Apple Standard Libraries
  - You can link with BSD Libraries in CW to aid the transition

Your Code and GCC

Apple Standard Libraries and MSL

- MSL Mach-O lives on top of Apple Standard Libraries
  - You can link with BSD Libraries in CW to aid the transition
- Panther adds wchar_t, wstring support
  - Based on Citrus project—heavily tested in NetBSD

Your Code and GCC

Apple Standard Libraries and MSL

- MSL Mach-O lives on top of Apple Standard Libraries
  - You can link with BSD Libraries in CW to aid the transition
- Panther adds wchar_t, wstring support
  - Based on Citrus project—heavily tested in NetBSD

- Apple libraries use global, not std::, namespace

Your Code and GCC

Apple Standard Libraries and MSL

- MSL Mach-O lives on top of Apple Standard Libraries
  - You can link with BSD Libraries in CW to aid the transition
- Panther adds wchar_t, wstring support
  - Based on Citrus project—heavily tested in NetBSD

- Apple libraries use global, not std::, namespace
- Apple STL doesn't have Metrowerks-style debug mode

Your Code and GCC

Apple Standard Libraries and MSL

- MSL Mach-O lives on top of Apple Standard Libraries
  - You can link with BSD Libraries in CW to aid the transition
- Panther adds wchar_t, wstring support
  - Based on Citrus project—heavily tested in NetBSD

- Apple libraries use global, not std::, namespace
- Apple STL doesn't have Metrowerks-style debug mode
- Apple libraries don't have Metrowerks extensions
  - e.g., FSSpec versions of POSIX file calls

Your Code and GCC

Compatibility features added to GCC

- Some #pragma support
- Metrowerks-style inline assembly

Optimizing Build and Turnaround Time

Optimizing Build and Turnaround

Enabling Xcode features

- Use gcc 3.3

Optimizing Build and Turnaround

Enabling Xcode features

- Use gcc 3.3
- Remove .mch and cpp-precomps

Optimizing Build and Turnaround

Enabling Xcode features

- Use gcc 3.3
- Remove .mch and cpp-precomps
- Make sure to have a precompiled prefix file
  - Include all framework headers
  - The more project headers you use, the faster it is

Optimizing Build and Turnaround

Enabling Xcode features

- Use gcc 3.3
- Remove .mch and cpp-precomps
- Make sure to have a precompiled prefix file
  - Include all framework headers
  - The more project headers you use, the faster it is
- Enable Predictive Compilation (Prefs/Building)

Optimizing Build and Turnaround

Enabling Xcode features

- Use gcc 3.3
- Remove .mch and cpp-precomps
- Make sure to have a precompiled prefix file
  - Include all framework headers
  - The more project headers you use, the faster it is
- Enable Predictive Compilation (Prefs/Building)
- Enable Zero Link (Project Info/Styles)

Optimizing Build and Turnaround

Enabling Xcode features

- Use gcc 3.3
- Remove .mch and cpp-precomps
- Make sure to have a precompiled prefix file
  - Include all framework headers
  - The more project headers you use, the faster it is
- Enable Predictive Compilation (Prefs/Building)
- Enable Zero Link (Project Info/Styles)
- Enable Fix and Continue (Project Info/Styles)

Optimizing Build and Turnaround

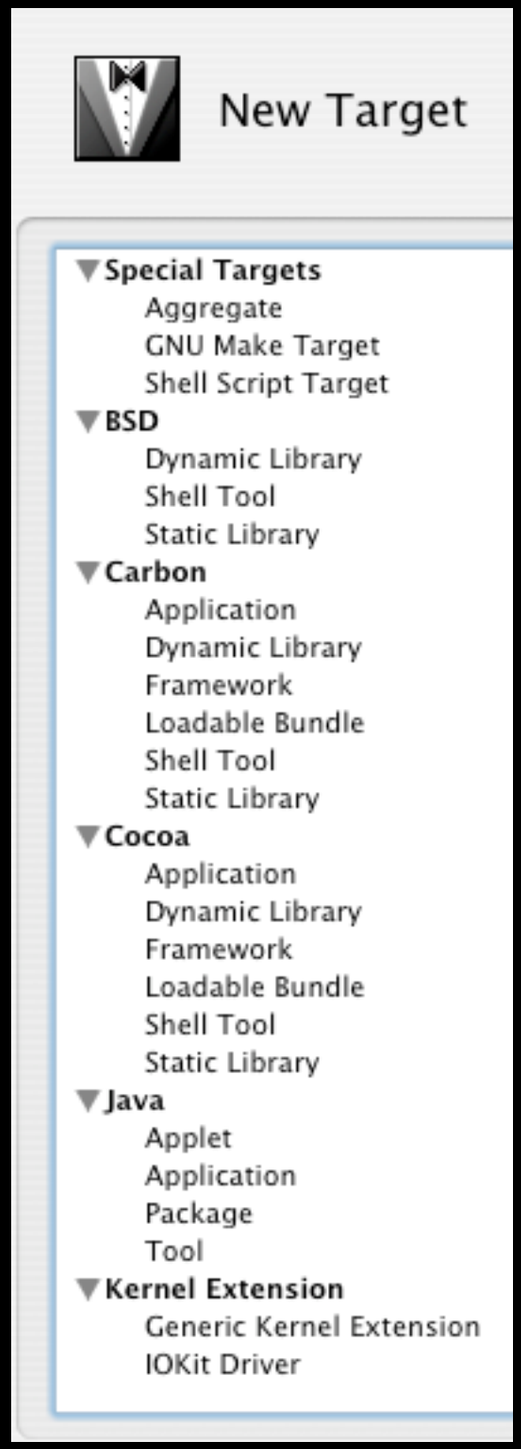
Enabling Xcode features

- Use gcc 3.3
- Remove .mch and cpp-precomps
- Make sure to have a precompiled prefix file
  - Include all framework headers
  - The more project headers you use, the faster it is
- Enable Predictive Compilation (Prefs/Building)
- Enable Zero Link (Project Info/Styles)
- Enable Fix and Continue (Project Info/Styles)
- Try Distributed Builds for big C++ projects

Packaging and Localizing Your Carbon Project

Packaging Your Target

- Pick the target stye when you define the target
- Most targets are packages
  - Application, framework, bundle
- See TN2015 for details



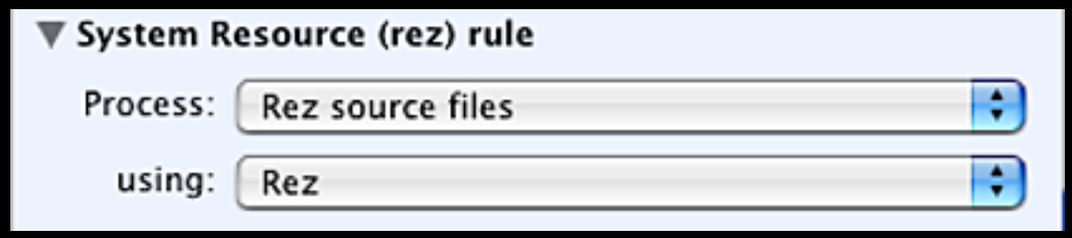
Packaging Your Target

Where does Xcode put my stuff?

- Executables go into Contents/Mac OS
- Resources go into Contents/Resources
- Target settings go into Info.plist
- There'll be more control in the 1.0 release

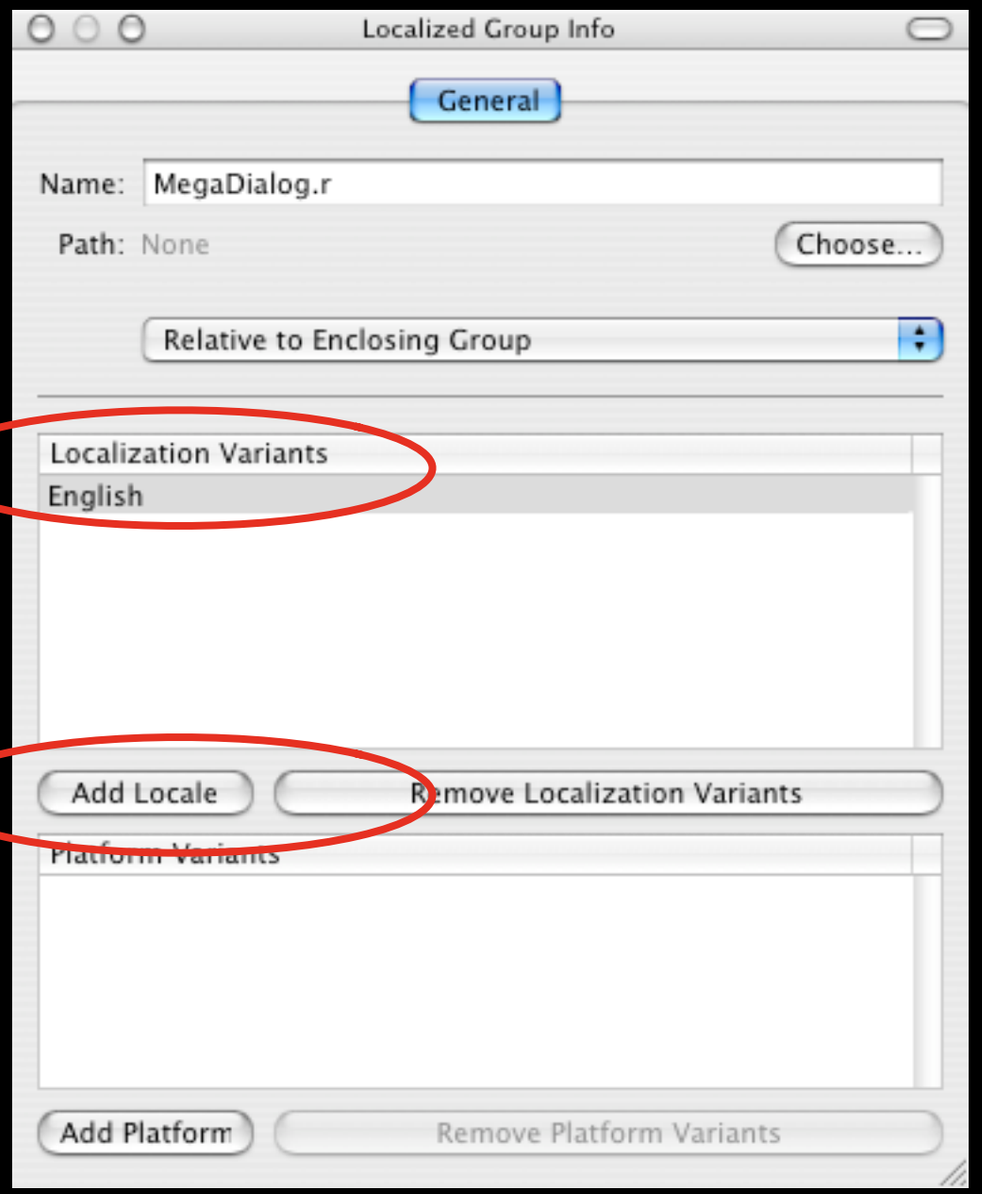
Working With Resources In Xcode

- Rez for .r files into .rsrc
- RezMerger for .rsrc, .ppob, and other resource files
- Interacts with Resourceror, Constructor, Interface Builder



Localizing Resources

- Mac OS X apps are multilingual
- Use the File inspector
- Add variants for languages you support



Interface Builder for Carbon

Interface Builder File Edit Classes Format Layout Tools Window Help Tue 10:16 AM

Window

Box

HIImage

First Name

Last Name

OK

Carbon-Controls

Text

Button

Bevel

StaticText

CheckBox

RadioButton

Radio1

Radio2

Item 1

Data Browser (Column View) Info

Attributes

Hilite Style: Fill

Focus Ring

Row Height: 22

Column Width: 0

Variable

Row Height

Column Width

Scroll Bars

Horizontal

Vertical

Selection

Drag Select

Only One

Reset Selection

Cmd Toggles

No Disjoint

Always Extend

Untitled – MenuBar

NewApplication File Edit Window

Untitled

Instances Images Nib

MenuBar

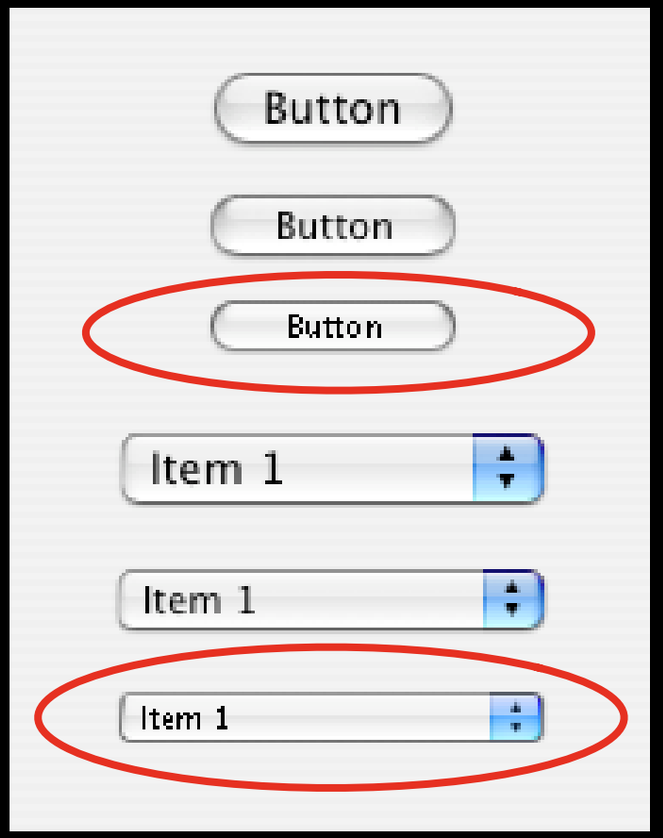
MainWindow

New for Carbon

- New supported Controls
  - HIImageView
  - HISearchField
  - HIScrollView
  - HITextView

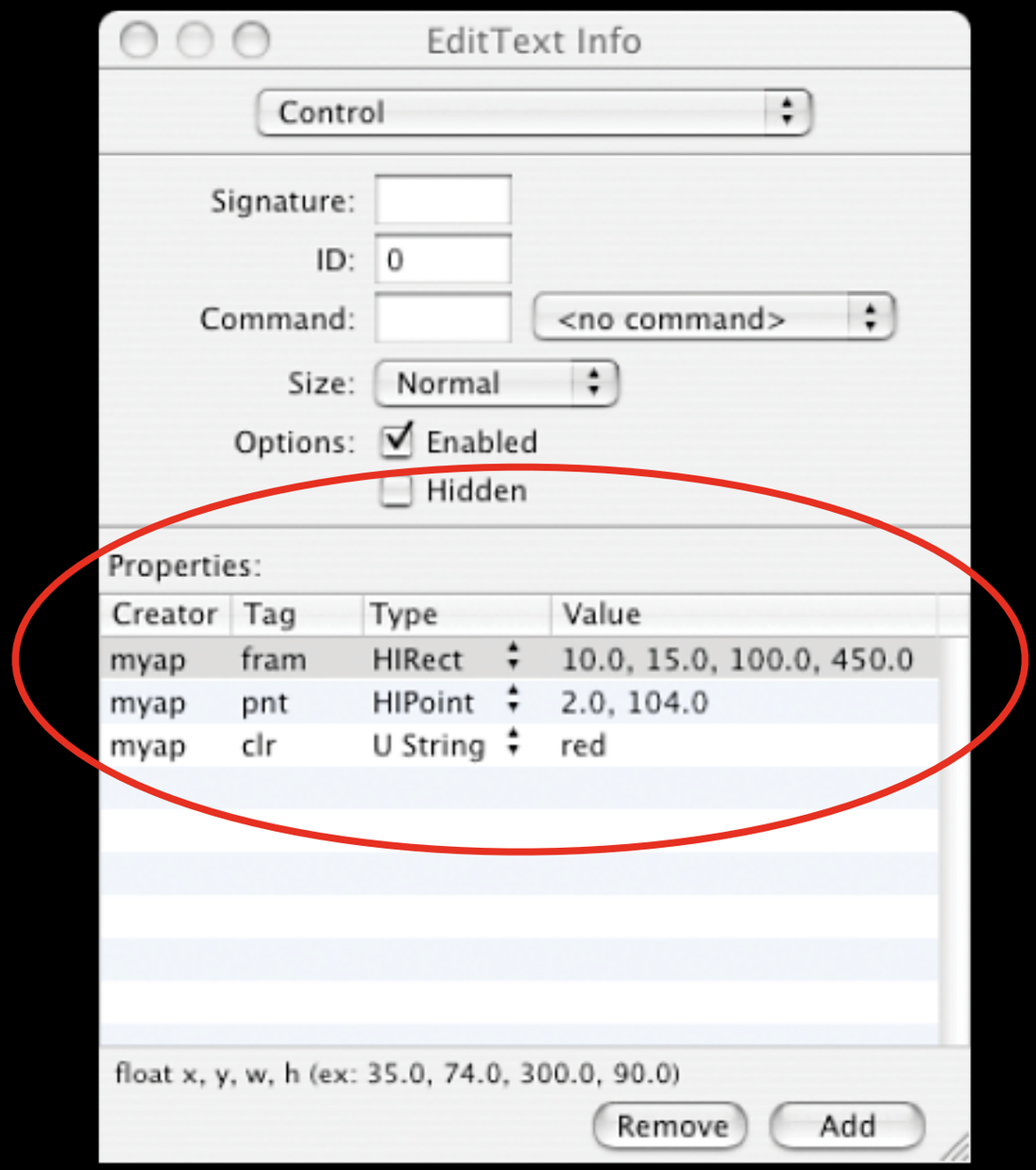
New for Carbon

- New supported Controls
  - HIImageView
  - HISearchField
  - HIScrollView
  - HITextView
- Mini Controls



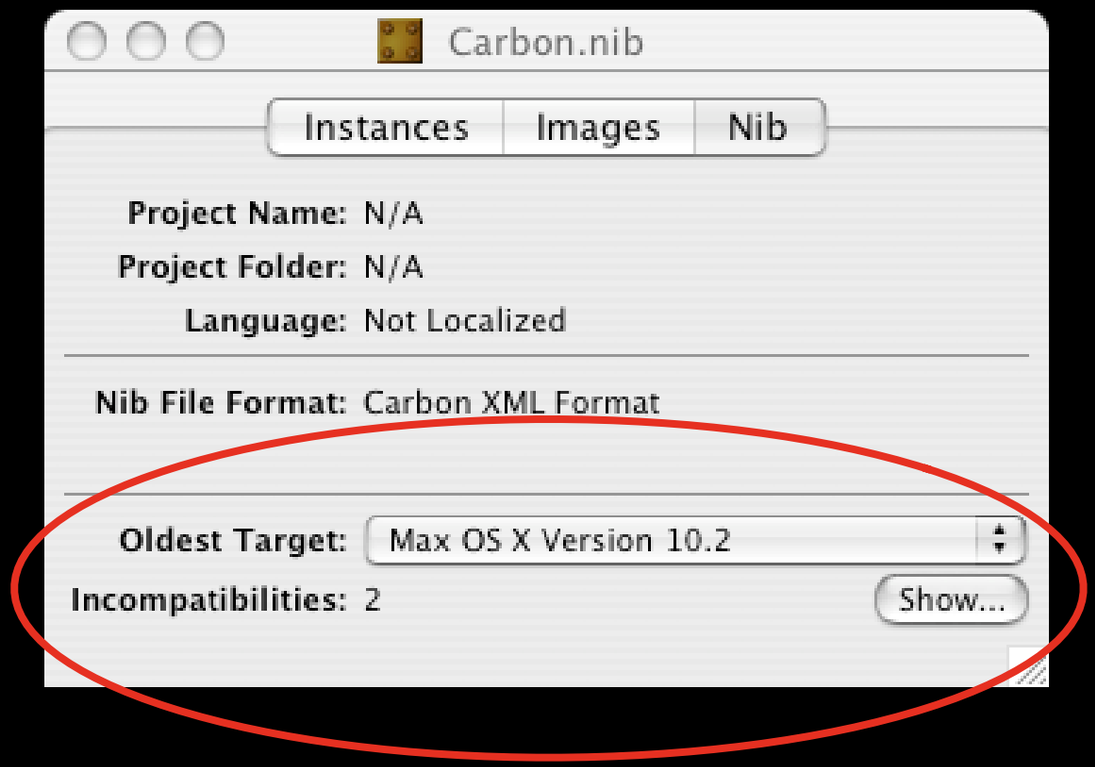
New for Carbon

- New supported Controls
  - HIImageView
  - HISearchField
  - HIScrollView
  - HITextView
- Mini Controls
- Control Properties



New for Carbon

- New supported Controls
  - HIImageView
  - HISearchField
  - HIScrollView
  - HITextView
- Mini Controls
- Control Properties
- Strong Nib file compatibility checking



Wrap Up

Xcode Is Built for Your Carbon App

- Imports CodeWarrior projects
- Builds C++ code well
- Packages and builds Mac OS X Carbon applications

How to Learn More

- Part of Xcode tools package
  - Use the Documentation window and search features
- Inside Mac OS X books
  “Inside Mac OS X: System Overview”
- Apple Developer Connection tools pages
  http://developer.apple.com/tools/
- CVS documentation
  http://www.cvshome.org/docs/
- Perforce documentation
  http://www.perforce.com/perforce/technical.html

Reference Library

Documentation

Mach-O Runtime Architecture

ADC Home > Documentation > Tools >

GNU C/C++/Objective-C Compiler

ADC Home > Documentation > Tools >

Moving Projects From CodeWarrior to Project Builder (WWDC-only distribution)

ADC Member Site > Download Software >

Integrating Carbon and Cocoa in Your Application

ADC Home > Documentation > Cocoa > Design Guidelines >

Reference Library

Documentation (Cont.)

Xcode Release Notes

ADC Home > Documentation > Tools >

Interface Builder Release Notes

ADC Home > Documentation > Tools >

Reference Library

Technical Notes

Platforms and Tools

ADC Home > Technical Notes >

Technical Q&As

Platforms and Tools

ADC Home > Technical Q&As >

Sample Code

Platforms and Tools

ADC Home > Sample Code >

Who to Contact

Godfrey DiGiorgi
Technology Manager, Development Tools
ramarren@apple.com

Development Tools Engineering Feedback
xcode-feedback@group.apple.com

To Post Bug Reports
http://developer.apple.com/bugreporter

Mailing Lists at Apple
http://lists.apple.com

http://developer.apple.com/wwdc2003/urls.html

Roadmap

FF009 Development Tools Feedback Let us know what you need	**North Beach** **Thur., 2:00pm**
308 Debugging Services for Mac OS X Debugging with Xcode and gdb	**Presidio** **Thur., 3:30pm**
310 Debugging and Tuning Carbon Apps Debugging and performance for Carbon apps	**Presidio** **Fri., 9:00am**
311 Software Testing Tools for Mac OS X Automate your application testing	**Russian Hill** **Fri., 10:30am**
423 How to Write a Modern Carbon App Getting to know carbon events and Nibs	**Nob Hill** **Fri., 2:00pm**

Q&A

Chris Espinosa
Developer Tools Team

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference